How To Install Node.js with NVM (Node Version Manager) on Server

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: *http://faq.asphosthelpdesk.com/article.php?id=226*

Introduction

If you already know what Node.js is what it's for and why it's cool, then skip straight to the installation directions. If you want to know a bit more about node and it's ecosystem read on.

For those who haven't heard node.js is, it is the hot new cool kid on the block in web application development. It lets you write web apps that use Javascript on both the server and the client, so you don't need to know multiple programming languages to program your website. It's also really good at handling real-time concurrent web applications, which makes it a great choice for a lot of modern webapps.

The downside though is that all these cool new features are really, really *new*. As a result, getting up and running with node.js isn't as easy as, say, getting WordPress up and running on your web server.

This is the first in a series of how to install, code in, and use node. Joyent, the team behind node.js, has been improving node.js at a frantic pace, to the point where there are multiple releases of the software every month. For the most part, they've done a pretty good job of keeping things compatible; the things you write for one version of node will work just as well in the next. But nonetheless, sometimes a particular node app will only work with one version of node. And you will need to upgrade or downgrade your node.js install in order to use it.

This used to be a pain, but the node community has come together and created a great solution that lets you easily manage all your node installations and change node versions whenever you feel like it. It's called NVM, or the Node Version Manager.

Installing Node.js on a server

You'll see some output fly by, and then nvm will be installed. You will see a line that says:

=> Close and reopen your terminal to start using NVM

It's not actually necessary to log out, we just need to make sure that the changes nvm made to your path are actually reflected, so just do:

```
source ~/.profile
```

Alternatively, run the command suggested in the output of the script. Now type:

```
nvm ls-remote
```

Should you see the error, `-bash: nvm: command not found` it may be because git is not installed.

Go ahead and install git and rerun the script:

```
apt-get install git
```

And you will be shown a list of all the available versions of node.js. You can always find out the latest stable release by heading to the node.js website, where it's printed in the center of the page. To install version 0.10.13 (the latest as of this writing) type:

```
nvm install 0.10.13
```

If you type:

```
node --version
```

You will now see that node v0.10.13 is installed and active. If you had an older node app that only works with node v0.8.16, and wanted to downgrade, then you would input:

```
nvm install v0.8.16
```

to install and switch to v0.8.16.

When you're done and want to switch back to v0.10.13, you can do so with nvm's use command:

```
nvm use v0.10.13
```

Nvm is great and makes switching between node versions easy and convenient. However, there's one caveat. If you type:

```
which node
```

you will see something interesting. Nvm installs node.js inside your user's home directory. This is fine for development, but if you want to actually host node applications, you don't want to install the latest new version of node via nvm and discover that you've inadvertently caused your production node app (which can be incompatible with the latest node.js) to stop working. It's best to install one copy of node globally so that other users can access it, and use nvm to switch between your development versions. To do this, run the following command (entering your user's password at the prompt):

```
n=$(which node);n=${n%/bin/node}; chmod -R 755 $n/bin/*; sudo cp -r
$n/{bin,lib,share} /usr/local
```

The above command is a bit complicated, but all it's doing is copying whatever version of node you have active via nvm into the /usr/local/ directory (where user installed global files should live on a linux server) and setting the permissions so that all users can access them.

If you ever want to change the version of node that's installed system wide, just do another nvm use vXX.XX.XX to switch your user's node to the version you want, and then re-run the above command to copy it to the system directory. To check that it works, become the root user and do another which command to make sure that node is now installed to /usr/local/bin:

```
sudo -s
which node
```

You should see:

```
/usr/local/bin/node
```

Congrats! Node.js is now installed and ready for use. Enjoy!